

Optimizing Mixing in Pervasive Networks: A Graph-Theoretic Perspective

Murtuza Jadliwala, Igor Bilogrevic, and Jean-Pierre Hubaux

LCA1, EPFL, Lausanne, Switzerland.
firstname.lastname@epfl.ch

Abstract. One major concern in pervasive wireless applications is location privacy, where malicious eavesdroppers, based on static device identifiers, can continuously track users. As a commonly adopted counter-measure to prevent such identifier-based tracking, devices regularly and simultaneously change their identifiers in special areas called mix-zones. Although mix-zones provide spatio-temporal de-correlations between old and new identifiers, pseudonym changes, depending on the position of the mix-zone, can incur a substantial cost on the network due to lost communications and additional resources such as energy. In this paper, we address this trade-off by studying the problem of determining an optimal set of mix-zones such that the degree of mixing in the network is maximized, whereas the overall network-wide mixing cost is minimized. We follow a graph-theoretic approach and model the optimal mixing problem as a novel generalization of the vertex cover problem, called the *Mix Cover (MC)* problem. We propose three bounded-ratio approximation algorithms for the MC problem and validate them by an empirical evaluation of their performance on real data. The combinatorics-based approach followed here enables us to study the feasibility of determining optimal mix-zones regularly and under dynamic network conditions.

1 Introduction

Recent advances in wireless and mobile computing technology have resulted in rapid proliferation and use of this technology for a variety of pervasive computing and data-sharing applications. Popular instances of this networking technology include vehicular and pervasive social networking systems and applications such as vehicular safety messaging [48,37], pervasive or local-area social networking [41,4,10], dating [1,2,33], personal safety [39] and micro-blogging [21]. Communication devices such as mobile phones (in personal networks) or wireless on-board computers (in vehicular networks) communicate with each other directly in a peer-to-peer fashion or with third-party service providers through an infrastructure such as a base station or road-side unit.

Users in such pervasive systems continuously face privacy risks, especially in terms of location privacy, from malicious eavesdroppers and curious service providers. Users' location information revealed as a result of this threat can be used by malicious parties to track their movements and preferences [20] or

to identify users and their availabilities by inferring their home/work locations [25], which can be later used for accomplishing malicious goals [3]. Third-party service providers, however, are generally trusted and claim to utilize the collected personal and location information to further enhance context-aware services but can inadvertently harm users' privacy if the collected data is improperly shared with commercial partners or leaked in an unauthorized fashion.

One widely adopted strategy to overcome the location privacy concerns in such pervasive systems, which is inspired by Chaum's seminal work on mix networks [13,14], is to regularly mix [35] or change [29,12] device identifiers including application, IP and device MAC addresses. Recently, non-IP networks such as cellular networks adopt a similar approach; they identify a subscriber's device with a Temporary Mobile Subscriber Identity or TMSI that changes every time the subscriber moves to a new geographical area.

In order to maximize anonymity, mixing or changing of user identifiers should occur in a spatiotemporal region, called *mix-zones* [8,9], where a group of nodes do not transmit any information (or identifiers); on leaving the mix-zone the communication resumes with a new identifier or pseudonym for each user or device. Mix-zones serve to mix or provide de-correlation between pseudonyms and device associations, which makes it difficult for an adversary to continuously track users by linking the device and its pseudonym. Let us focus on pervasive and mobile networking scenarios where users or devices generally move on a fixed (pre-defined) network of roads, for example, vehicular or pedestrian handheld networking scenarios. Earlier research has shown that mix-zones in such networks are most effective (in protecting privacy) when they are defined at points with higher input and output ports such as *road intersections* [11].

Although effective in improving the privacy of users, the pseudonym-change (or mix) operation is not free and induces a cost on the network (and its users), which is determined by factors such as the significance of the intersection to users and the network, traffic intensity at the intersection (both entering and leaving) and intersection context, for example, time-of-day. This cost is primarily due to the loss in communication due to routing disruptions [43] or silent periods [29] and the loss of computation resources such as energy due to the pseudonym change operation itself and its related side-effects.

This results in an interesting trade-off between the number of mix-zones that can be deployed on the road network for privacy enhancement and the resulting cost due to such a deployment. An ideal situation from the privacy perspective, although infeasible from the cost point-of-view, is to deploy a mix-zone at each and every intersection of the road network under consideration. Such a deployment of mix-zones is trivial in theory, but difficult to realize and sustain in practice due to the resulting costs. A more realistic and feasible goal would be to maximize the coverage (of roads) of the deployed mix-zones, and hence the privacy provided by them, and to minimize the associated costs due to such a deployment. Moreover, the goal is not only to determine such an optimal and cost-efficient placement of mix-zones but also to study if there are algorithms that can find such a solution efficiently (in computation time and

space). This is because, as pseudonym change costs at intersections are highly dynamic and depend on factors such as intersection context and traffic intensity that continuously change over time, there is a need to regularly determine the optimal and most cost-efficient set of mix-zones. In order to design efficient algorithms for the above optimization problem, a thorough theoretical analysis of the problem from a combinatorial perspective is first required.

In this paper, we model the problem of optimal mix-zone placement as a graph-based optimization problem where roads are represented by graph edges and intersections by vertices. Vertices are weighted based on the cost (per device) of mix-zone placement at each vertex and edges are weighted based on the demand or traffic intensity of the corresponding road in *each direction*. The problem of optimal mix-zone placement - we refer to as the *Mix Cover problem (MC)* - is then to determine a set of intersections (or vertices) for mix-zone placement, such that all the roads in the network are associated with at least one mix-zone and the overall cost of the mix-zone placement is minimized. The mix cover problem nicely models the mix-zone placement problem in pervasive networks and is a generalization of the well-known Vertex Cover (VC) problem [32], and a special case of the Facility Terminal Cover (FTC) problem [47]. To the best of our knowledge, this generalization, and specifically in the setting of pervasive networks, has not been addressed in the literature. We show that the mix cover problem is a combinatorially hard problem and propose three bounded-ratio approximation algorithms for the same. The first algorithm is based on a linear programming relaxation of an Integer Program (IP) formulation of the problem, whereas the remaining two algorithms take advantage of the “divide and conquer” strategy of [47] which was used to solve the FTC problem. We analytically study the solution quality and running-time guarantees of the algorithms by deriving their worst-case approximation ratio and running-time, respectively. Finally, we perform an extensive comparative analysis of the proposed algorithms by evaluating them on real US road-traffic data.

2 Background and Related Work

In the following section, we provide a short overview of concepts in complexity theory and combinatorial optimization used throughout the paper and outline other research efforts on the mix-zone placement problem.

2.1 Preliminaries: Combinatorial Hardness and Approximations

A decision problem $S \subseteq \{0, 1\}^*$ is said to have an *efficiently verifiable proof system* if there exists a polynomial p and a polynomial-time verification algorithm V such that the following two conditions hold:

- Completeness: For every $x \in S$, there exists y of length at most $p(|x|)$ such that $V(x, y) = 1$.
- Soundness: For every $x \notin S$ and every y , it holds that $V(x, y) = 0$.

The class NP is the class of decision problems that have an efficiently verifiable proof system. A polynomial-time computable function f is called a *Karp-reduction* of S to S' (in other words, S is Karp reducible to S') if, for every x , it holds that $x \in S$ if and only if $f(x) \in S'$. A set S is *NP-complete* if it is in NP and every set in NP is Karp-reducible to it. A set S is *NP-hard* if every set in NP is Karp-reducible to it, but its membership within NP is not known. It is not known whether every problem in NP can be efficiently (in polynomial time) solved. But, if any single problem in the set of NP -hard problems can be solved efficiently, then every problem in NP can also be solved efficiently. Thus, NP -hard problems are considered “harder” than NP problems in general, and are believed to have no polynomial-time exact solutions. Algorithms for such hard problems, also called *optimization problems*, that run in polynomial time and produce a near-exact or sub-optimal solution are called *approximation algorithms*. Approximation algorithms that can guarantee that the solution output by them can be no more (if minimization problem) or less (if maximization problem) than a factor σ times the optimal solution is called a σ -*approximation algorithm* for that problem. More details on these topics can be found in [22,24].

2.2 Mix-Zone Placement Problem

The concept of using mix-zones in road networks, as a means to improve the location privacy of the mobile devices, has been proposed in [29,11,18]. Freudiger et al. [19] were the first to study and formulate the problem of optimal mix-zone placement in road networks. Here, the authors measure the *effectiveness* of mixing by measuring the probability of error of an adversary in correctly assigning exiting flows to their corresponding entering flows at a mix-zone. By using linear programming, they determine an optimal set of mix-zones that maximize the overall mixing effectiveness. In contrast, our model and solution is more general where we study the trade-off between maximizing the *coverage* of mix-zones and minimizing their *deployment cost*.

In another related effort, Alpcan and Buchegger [5] use game theory to model the attack and optimal defense strategies of the adversary and users in vehicular networks. Humbert et al. [31] also study the problem of optimal mix-zone placement from a game-theoretic perspective. They model the problem of mix-zone placement as a game between mobile users who want to protect their privacy and a local adversary who wants to track them by strategically placing eavesdropping stations. Here, the authors focus on deriving mix-zone deployment strategies locally at each intersection, whereas in our work, we study the problem of achieving a globally optimal deployment strategy. Palanisamy et al. [38] propose a framework and a suite of algorithms for mix-zone construction, which considers the inherent characteristics of road networks. Similar to earlier results, these mix-zone deployment strategies protect against specific adversarial attacks and only consider local intersection parameters for mix-zone deployment.

We extend the state of the art in optimal mix-zone deployment as follows. First, we study the problem of optimal mix-zone deployment from a global (network-wide) perspective. Moreover, our model and assumptions are generic

enough to include other privacy metrics [44,45], in addition to the basic mix-zone coverage guarantee. Second, the analytical results obtained in this paper shed light on the feasibility of optimally deploying mix-zones in dynamic real-time road-network settings autonomously by mobile devices. Finally, the results outlined in this paper are also significant from the combinatorics viewpoint, as the generalization of the VC problem studied here has not been discussed before in the literature.

3 Problem Statement

3.1 System Model

Consider a wireless and pervasive mobile networking system where users (or vehicles) carry wireless communication devices that can either communicate with each other in a peer-to-peer fashion or through infrastructure-based base station(s). Examples of such networking systems include, but are not limited to, wireless peer-to-peer mobile-phone based pervasive social networking platforms such as Nokia Instant Community (NIC) [41] and vehicle-based wireless communication systems or VANETs [23]. Each mobile device in the network includes some identifying information or pseudonym in its communication, such as a MAC address or an application-level identifier, which is used for identifying the device and for routing communications within the network [40].

In order to prevent trivial tracking by an eavesdropping adversary, wireless devices regularly change their identifiers or pseudonyms. Various techniques for privacy protection, which use multiple pseudonyms or identifiers, have been studied in the literature [9,35,11,12]. In order to prevent trivial linkability of old and new pseudonyms, devices must coordinate their pseudonym changes, in space and time, with other devices, in order to achieve spatial and temporal de-correlation. Such regions for achieving spatial and temporal de-correlation of devices and (old and new) pseudonyms are also referred to as *mix-zones* [9]. In a mix-zone, spatial de-correlation is achieved by mobile device(s) changing their pseudonyms in a coordinated fashion whereas temporal de-correlation is achieved by either remaining silent for a short period of time [29], by encrypting communications after pseudonym change [18], or by means of a mobile proxy [42]. Mix-zones can be *passive* or *active*, depending on the actions taken by the devices immediately after the pseudonym change operation [31]. We assume that an off-line Certification Authority (CA), run by an independent trusted third-party, loads the mobile devices with a set of pseudonyms prior to deployment.

Road *intersections* are considered to be good spots for mix-zone deployment (and coordinated pseudonym change operations) as they provide optimal spatio-temporal de-correlation, as also observed in [19,31]. It is clear that mix-zones incur a communication overhead [43] and thus must be carefully placed (with appropriate parameters [30]) in order to reduce the cost induced on the end-users and to provide high location privacy (or high user-identifier de-correlation). The cost of deploying a mix-zone at any intersection can be a weighted sum of various factors, including the extra resource requirements of devices for mixing and the

resulting communication disruption due to mixing at that intersection. We do not quantify these parameters in this work, but we can use existing results in the literature for representing these costs [43,31].

We assume that all the intersections, over the area under consideration, are connected with each other by a network of roads. Each road can be used to reach either one of the intersection that it connects, i.e., there is a two-way movement of users (or devices, vehicles, etc.) on the road. The *demand* for an intersection on a road is the average number of users using the road to reach that particular intersection. Thus, each road has two demands, one for each intersection connected by the road. Accordingly, unidirectional roads have just one demand, i.e., the one in the direction of the intersection; the other demand is zero. For simplicity, we assume that any two intersections are connected only by a single road; multiple roads between any two intersection can be combined into a single road by simply adding their respective demands.

3.2 Privacy Requirement

Given the system model discussed above, we want to investigate the problem of determining the most effective and cost-efficient mixing strategy in large pervasive networking scenarios. In other words, we address the problem of determining an optimal selection of intersections for mix-zone deployment such that all the roads in the network are covered and the *overall* cost due to mixing is minimized. We say that a road is *fully-covered* if and only if *both* the end points (intersections) of the road have mix-zones deployed on it, i.e., there is mixing at both intersections of the road. A network is said to be *fully-covered* (or has a *full cover*) if and only if all the roads in the network are fully-covered.

It is easy to see that in the system model discussed above, a full covering of the network can only be achieved if and only if all the intersections in the network are selected for mixing or mix-zone deployment. Such a mixing or full covering strategy is not only trivial but also ideal from the privacy viewpoint. But from a cost perspective, such a covering may be difficult to achieve in practice due to the network size or infeasible due to the overall cost of mixing at the intersections.

Let us now define a more general version of the full cover described above, called *mix cover*. A network is said to be *mix covered* if and only if each of the roads in the network have at least one of its intersections where a mix-zone is deployed. A fully-covered network is also mix covered and some of the roads in a mix covered network may be fully-covered, i.e., both the intersections of a road may have mix-zones deployed. From the privacy perspective, a mix covered network can *guarantee* that any user (or device) traversing the road network can traverse *at most two* roads (or *at most one intersection*) without encountering a mix-zone. From the practical standpoint, a mix cover is a reasonable mixing strategy for most deployment scenarios and adversarial models. We focus on the problem of determining a cost-efficient mix cover by modeling it as a *graph-based optimization problem*, as discussed next.

3.3 Graph-theoretic Framework and the Mix Cover (MC) Problem

Let us represent the road network described above by an *undirected graph* $G \equiv (V, E, w, d)$. There exists a vertex $v \in V$ corresponding to each intersection in the road network and $|V| = n$ is the total number of intersections (vertices¹) in the area of the road network under consideration. Each road connecting any two intersections u and v is represented by an edge $e \equiv (u, v) \in E$, where E is the set of all edges (or roads) and $|E| = m$ is the total number of roads (edges). There exists only a single edge (u, v) between any two pair of vertices u and v in G . Given the undirected graph G , let $w : V \rightarrow \mathbb{R}^+$ be the *cost function* that assigns a positive cost to each vertex. The cost at each vertex represents the average cost (per user) of mix-zone deployment (or mixing) at that intersection; the higher the cost, the higher the amount of communication and device resources spent by each user for mixing at that intersection is. We represent by w_u the cost of a vertex $u \in V$. Let $d : E \rightarrow (\mathbb{R}^+, \mathbb{R}^+)$ be the *demand function* that assigns a pair of positive demands to each edge where each demand value in the pair represents the demand for a particular vertex connected by the edge. This demand pair could signify, in the case of vehicular (or pedestrian) networks, the average traffic (or pedestrian) intensity on the road in each direction. For any edge $e(u, v) \in E$, we represent the demand as $d_e = (d_e^u, d_e^v)$, where d_e^u, d_e^v is the demand on edge e for intersections u and v , respectively. The value of $d_e^u = 0$ if u is not one of the end points of the edge e .

Given the above graph representation of the road network, we are interested in the problem of efficiently determining the *optimal* mix cover of the network. Each vertex chosen in the mix cover should be able to handle the demands from all the edges it covers. In other words, each intersection should be able to accommodate even the largest demand made at it; we refer to this ability of each intersection as the *capacity* of the mix-zone at that intersection. The capacity at a vertex is zero if there is no mix-zone at that vertex. The optimality criteria is based on an assignment of capacities to vertices or intersections such that the demands of all edges are met and the overall cost minimized. Formally, we can represent the problem of determining the optimal mix cover, referred by us as the *Mix Cover (MC) problem*, in the graph $G \equiv (V, E, w, d)$ as a generalization of the *Vertex Cover (VC)* problem. VC is a fundamental problem in graph theory and a vertex cover of an undirected graph $G \equiv (V, E)$ is a subset of vertices $V_C \subseteq V$ which contains at least one vertex of all the edges in E and the VC problem is to determine a vertex cover V_C of the smallest cardinality. The VC problem is *NP-hard* and the decision version of the same is known to be *NP-complete* [32]. The Mix Cover (MC) problem can be formally defined as:

Problem 1. Given an undirected graph $G \equiv (V, E, w, d)$, where w is the cost function associated with the set of vertices and d is the demand function associated with the set of edges, as discussed above, determine a subset $V_{MC} \subseteq V$ and

¹ Readers should note that from this point on we will use “vertex” and “intersections” (similarly, “road” and “edge”) interchangeably and the intended meaning will be implicit from the context.

a capacity $c(v)$ for each vertex $v \in V_{MC}$ such that for each edge $e \equiv (u, v) \in E$ at least one of the vertices u and v is in V_{MC} and associated with a capacity $c(u) \geq d_e^u$ and $c(v) \geq d_e^v$ respectively, and the total weighted cost, $\sum_{x \in V_{MC}} c(x)w_x$, of all vertices in V_{MC} is minimized.

Thus, given graph $G \equiv (V, E, w, d)$ of the road network, the MC problem determines a mix cover of the network such that the overall (network-wide) weighted cost of the mix cover is minimized. The total intersection cost at each intersection v is the intersection mixing cost w_v times the capacity $c(v)$ at v . The capacity at any intersection v is at least the maximum demand at that intersection from all roads covered by it. The overall (network-wide) weighted cost of the mix cover is the sum of all the total intersection costs at each intersection in the mix cover. Figure 1 shows one such feasible solution.

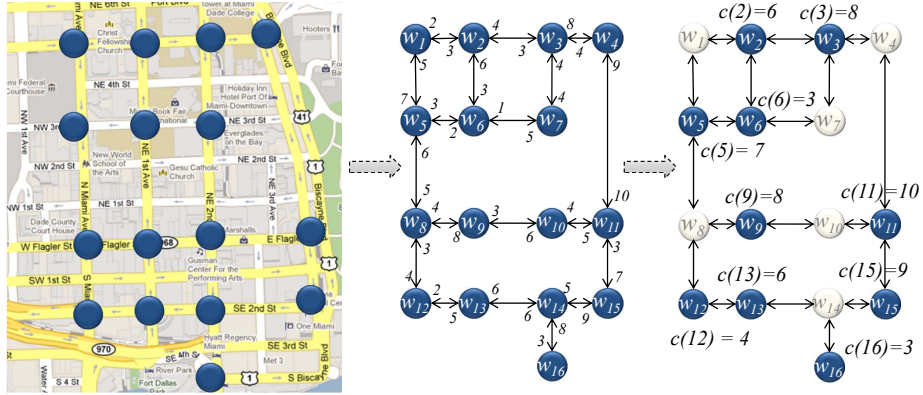


Fig. 1. Mix Cover example on downtown Miami (FL). On the left, the dark circles indicate all intersections where mix-zone placement is possible. The graph representation is shown in the middle and a feasible mix cover is shown on the right, where the dark circles are included in the solution and the shaded circles are not.

The MC problem is very similar to another generalization of the VC problem called the Facility Terminal Cover (FTC) problem [47,28], but there is an important difference between the two problems. Given a graph $G \equiv (V, E, w, d)$, where $w : V \rightarrow \mathbb{R}^+$ and $d : E \rightarrow \mathbb{R}^+$ (denoted as w_v and d_e for vertex v and edge e , respectively), the FTC problem is to find a set $V_{FTC} \subseteq V$ and a capacity $c(v)$ for each vertex $v \in V_{FTC}$ such that for each edge $e \equiv (u, v) \in E$ at least one of the vertices u and v is in V_{FTC} and associated with a capacity $c(u) \geq d_e$, and the total weighted capacity $\sum_{x \in V_{FTC}} c(x)w_x$ is minimized. As we can see from the FTC problem definition, the assumed graph model assigns only a *single* demand value to an edge and so the selected capacity for covering any edge depends only on the demand value for this same edge. This is different from the MC problem where each edge has two demand values and the selected capacity for covering any edge depends on the (demand value associated with the) vertex that is used

to cover that same edge. The FTC problem can be considered as a special case of the MC problem, i.e., the MC problem reduces to the FTC problem when both the demand values are equal for all the edges in the graph. The formulation and algorithms of the FTC problem cannot be directly used to solve the much more general MC problem; although we will use one of the solution strategy [47] of the FTC problem for solving the MC problem.

There is another generalization of the VC problem called the minimum Generalized Vertex Cover (GVC) problem [27]. In GVC, contrary to VC, an edge incurs a cost (or demand, as in our case) depending on the *number* of its vertices that belong to the solution. Once again, such a generalization of the VC is different from the one that we are interested in, because in our case the demand incurred by the edge does not depend on the number of its vertices in the solution rather on which vertex is included in the solution. To the best of our knowledge, this is the first paper to model and study the problem of optimal mixing or mix-zone placement in pervasive networks as a unique generalization of the VC problem, which we believe has not been studied before.

4 Algorithms and Combinatorial Results

Let us first analyze the combinatorial hardness of the MC problem. We state the following theorem for the hardness of the MC problem.

Theorem 1. *The MC problem is NP-hard.*

Theorem 1 is straightforward, as we can easily reduce any instance of the weighted vertex cover problem to an instance of the MC problem in polynomial time. This can be done by defining a simple demand function for the graph instance of the weighted vertex cover problem as $d_e \equiv (d_e^u = 1, d_e^v = 1), \forall e$, where $e \equiv (u, v)$ is an edge of the graph instance. Thus, as the weighted vertex cover is NP-hard, we can claim that the MC problem is also NP-hard. The MC problem also seems difficult to approximate and we do not believe it has a Polynomial-Time Approximation Scheme (PTAS). This is because the VC problem itself, which is considered to be much simpler than the MC problem, is not believed to have an approximation ratio within 1.3606 unless $P = NP$ [16]. In the following sections, we outline two approximation strategies for the MC problem. The first is based on a linear programming formulation of the problem, whereas the other two algorithms employ a “divide and conquer” strategy by utilizing the round and group approach for solving the FTC problem [47].

4.1 Linear Programming Algorithm

We first formulate the MC problem as an Integer Program (IP), more specifically a 0-1 Program. Let z_e^v be a binary decision variable for each edge e and its corresponding vertex v which indicates whether the vertex v is included in the mix cover (solution) to cover edge e or not, i.e., $z_e^v = 1$ if edge e is covered by vertex v and $z_e^v = 0$ if not. Let x_v be the decision variable indicating the capacity

and w_v indicate the cost of each vertex $v \in V$. Then, the IP formulation of the MC problem can be obtained as follows:

$$\begin{aligned} & \min \sum_{v \in V} w_v x_v \\ & \text{subject to } z_e^u + z_e^v \geq 1, \forall e \equiv (u, v) \in E \\ & \quad x_v \geq z_e^v d_e^v, \forall v \in V, e \in E \\ & \quad z_e^v \in \{0, 1\}, \forall v \in V, e \in E \end{aligned}$$

Now, solving an Integer Program is a well-known hard problem [32]. Fortunately, efficient (polynomial time) techniques [6] exist for solving a Linear Program (LP) relaxation of the Integer program. If the LP relaxation has an integral solution then that can also be the solution to the above IP. In general, solving the LP relaxation of the problem can give a fractional feasible solution, from which a feasible (and possibly non-optimal) solution to the above IP can be obtained. The LP relaxation of the problem is as shown below:

$$\begin{aligned} & \min \sum_{v \in V} w_v x_v \\ & \text{subject to } d_e^v x_u + d_e^u x_v \geq d_e^u d_e^v, \forall e \equiv (u, v) \in E \\ & \quad x_v \geq 0, \forall v \in V \end{aligned}$$

Let $(\bar{x}, \{\bar{z}_e | \forall e \in E\})$ be an optimal solution to the above LP formulation, where $z_{e,i} = \frac{x_i}{d_e^i}$ is the entry of the vector \bar{z}_e representing the value of the decision variable corresponding to vertex i (to cover edge e), and x_j is the j^{th} entry of \bar{x} and represents the capacity value at the vertex j . The value of $z_{e,i} = 0$ if i is not a vertex in edge e . We can see that any optimal solution $(\bar{x}, \{\bar{z}_e\})$ produced by solving the above LP is a feasible fractional solution to the MC problem. It is also clear that an optimal solution OPT to the MC problem is always a feasible solution to the above LP formulation. Thus, the above LP relaxation for the MC problem is correct. Based on this, we can prove the following bound on the approximation quality for the MC problem.

Theorem 2. *There exists a polynomial time 2-approximation for MC.*

For conciseness, the proof of this theorem has been moved to the Appendix. Theorem 2 shows that a constant ratio approximation is possible for the MC problem. Algorithms for linear programming, such as the simplex algorithms [15], are efficient in practice with a polynomial (in number of constraints) number of iterations, excluding the number of arithmetic operations [36]. But, Klee and Minty [34] showed that the number of iterations performed by some variants of the simplex can be exponential. Moreover, there is always a possibility, depending on the demand values, of the method producing an unbounded or an infinite number of solutions. To overcome these problems, we take advantage of a deterministic linear-time (in number of edges) approach for FTC proposed by Xu et al. [47], as discussed next.

4.2 “Divide and Conquer” Algorithms

In their algorithm, Xu et al. divide the input graph instance into multiple subgraphs by first rounding the edge demands and then grouping them based on the rounded edge values. They then apply the Weighted Vertex Cover (WVC) algorithm on each subgraph to obtain the solution to the FTC problem on the input graph. They show that their algorithm produces a 8-approximation when a 2-approximation algorithm [7,26] is used for WVC.

One of the main differences between FTC and MC is that in FTC the input graph instance has all edges with a single demand value, whereas in the MC problem, each edge has two demands, depending on the vertex chosen to cover the edge. Moreover, the MC problem is not directly reducible to the FTC problem unless the two demand values for each edge are equal. Below we outline two algorithms for solving the MC problem; they utilize the round and group strategy of [47]. In order to take advantage of their approach to solve the MC problem, we first need to transform the input graph instance so that all edges have equal demand values. Based on how this transformation is done, we will later see that the overall solution quality is accordingly influenced.

Largest Demand First (LDF) Algorithm In our first, and more straightforward approach, we transform an input instance of the MC problem from $G \equiv (V, E, w, d)$ to $G' \equiv (V, E, w, d')$ such that, for each edge, both the new demands of the edge are equal and with value equal to the larger of the two original demand values. The intuition behind such a transformation is that if a vertex is able to cover the larger demand, then it will definitely be able to cover any demand smaller or equal to the larger demand. Then, the final demand values of the edges in the new graph instance G' are *rounded* off to the closest power of 2 of the larger demand value chosen in the previous step. Lemma 1 shows that any solution of the MC problem on such a transformed version (G') of the original graph is also a feasible solution for the MC problem on original graph (G). After obtaining G' , it is first divided into subgraphs (G_k) based on the rounded edge demands (2^k), with each subgraph containing only edges of the same demand value. A known minimum WVC algorithm (such as [7,26]) is then used to obtain the minimum weighted vertex cover for each subgraph G_k . The mix cover is finally obtained by combining solutions from each of the individual subgraphs in the previous step. The LDF algorithm is outlined in Algorithm 1.

Lemma 1. *Any solution to the MC problem on the transformed graph instance $G' \equiv (V, E, w, d')$ is also a feasible solution to the MC problem on the original graph instance $G \equiv (V, E, w, d)$. Moreover, $OPT(G') \leq 2\alpha OPT(G)$, where $OPT(\cdot)$ is the optimal solution and $\alpha = \max\{|d_e^u - d_e^v| \mid \forall e \equiv (u, v) \in G\}$, i.e., the maximum difference, over all edges, between the two edge demand values.*

We have the following result for the solution quality and running time of LDF.

Theorem 3. *The LDF algorithm is a linear time (in terms of the number of edges and vertices), $4\alpha\beta$ -approximation algorithm for the MC problem, where*

Algorithm 1: Largest Demand First (LDF) Algorithm

```

input : A graph  $G \equiv (V, E, w, d)$ .
output: A mix cover  $S_{MC} \equiv (v, c(v))$  of  $G$ , where  $v \in V$  and  $c(v)$  is the capacity assigned
        to  $v$ .
for all  $e \equiv (u, v) \in E$  do
     $d'_e \equiv (d'^u_e = \max\{d^u_e, d^v_e\}, d'^v_e = \max\{d^u_e, d^v_e\})$ ;
    if  $2^{k-1} \leq d'^u_e = d'^v_e \leq 2^k$  then
         $d'_e \equiv (d'^u_e = 2^k, d'^v_e = 2^k)$ ;
    end
end
Let  $G' \equiv (V, E, w, d')$ ;
Let  $G_k \equiv (V_k, E_k, w)$  be a subgraph of  $G'$  induced by edges  $E_k = \{e \in E \mid d'_e = 2^k\}$ ;
for all  $G_k$  do
    if  $V_k \neq \phi$  then
         $S_{G_k} = \text{WVC-2Approx}(G_k \equiv (V_k, E_k, w))$ ;
    else
         $S_{G_k} = \phi$ ;
    end
end
 $S_{MC} = \phi$ ;
for all  $S_{G_k}$  such that  $S_{G_k} \neq \phi$  do
     $c(v) \leftarrow \max\{2^k \mid \forall k \text{ s.t. } v \in S_{G_k}\}$ ;
     $S_{MC} \leftarrow (v, c(v))$ ;
end

```

$\beta > 1$ is the approximation ratio of the minimum WVC algorithm used and α is as defined in Lemma 1.

The proof of Lemma 1 and Theorem 3 can be found in the Appendix. Now, let us present a second solution strategy based on a transformation that chooses the smaller of the two demand values.

Smallest Demand First (SDF) Algorithm In the LDF algorithm, we transform the input graph instance into an instance where the smaller edge demand is replaced by the larger one. This guarantees that each edge has the same (and a single) demand value and that the mix cover of such a transformed instance is also a feasible mix cover of the original instance. In practice, it is clear that such a strategy will produce a highly sub-optimal solution because there may be vertices in the final solution that may cover edges with much lower actual demand values. In order to overcome this issue, we propose another strategy for solving the MC problem, called the Smallest Demand First (SDF) algorithm.

This SDF algorithm, as outlined in Algorithm 2, consists of three phases. In the first phase, in contrast to the LDF algorithm, we transform the input graph instance $G \equiv (V, E, w, d)$ of the MC problem into an instance $G'' \equiv (V, E, w, d'')$ where the larger edge demand is now replaced by the smaller one. In this phase, an additional task during edge demands transformation is to remember the largest demand (d^v_{max}) to be covered at each vertex. In the second phase, similar to the LDF algorithm, we use the round and group strategy to obtain a mix cover for the transformed instance. In the final phase, we assign capacities to the vertices based on the output of the previous phase and the largest demand

Algorithm 2: Smallest Demand First (SDF) Algorithm

```

input : Graph  $G \equiv (V, E, w, d)$ .
output: Mix cover  $S_{MC} \equiv (v, c(v))$  of  $G$ , where  $v \in V$  and  $c(v)$  is the capacity assigned to  $v$ .
for all  $v \in V$  do
  |  $d_{max}^v = 0$ ;
end
for all  $e \equiv (u, v) \in E$  do
  | if  $d_e^u > d_{max}^u$  then
  | |  $d_{max}^u = d_e^u$ ;
  | end
  | if  $d_e^v > d_{max}^v$  then
  | |  $d_{max}^v = d_e^v$ ;
  | end
  |  $d_e'' \equiv (d_e''^u = \min\{d_e^u, d_{max}^u\}, d_e''^v = \min\{d_e^v, d_{max}^v\})$ ;
  | if  $2^{k-1} \leq d_e''^u = d_e''^v \leq 2^k$  then
  | |  $d_e'' \equiv (d_e''^u = 2^k, d_e''^v = 2^k)$ ;
  | end
end
Let  $G'' \equiv (V, E, w, d'')$ ;
Let  $G_k \equiv (V_k, E_k, w)$  be a subgraph of  $G''$  induced by edges  $E_k = \{e \in E \mid d_e'' = 2^k\}$ ;
for all  $G_k$  do
  | if  $V_k \neq \emptyset$  then
  | |  $S_{G_k} = \text{WVC-2Approx}(G_k \equiv (V_k, E_k, w))$ ;
  | else
  | |  $S_{G_k} = \emptyset$ ;
  | end
end
 $S_{MC} = \emptyset$ ;
for all  $S_{G_k}$  such that  $S_{G_k} \neq \emptyset$  do
  |  $c(v) \leftarrow \max\{2^k \mid \forall k \text{ s.t. } v \in S_{G_k}\}$ ;
  |  $S_{MC} \leftarrow (v, c(v))$ ;
end

```

d_{max}^v determined in the first phase. Lemma 2 gives the relationship between the MC problem on the transformed version G'' and the original graph G .

Lemma 2. $OPT(G'') \leq \frac{2}{\alpha} OPT(G)$, where $G'' \equiv (V, E, w, d'')$ is the shortest demand first transformation of the original graph instance $G \equiv (V, E, w, d)$, $OPT(\cdot)$ is the optimal solution of the MC problem on the input graph instance and $\alpha = \max\{|d_e^u - d_e^v| \mid \forall e \equiv (u, v) \in G\}$, i.e., the maximum difference, over all edges, between the two edge demand values in the original graph instance.

It is easy to see that a feasible solution for the MC problem on G'' may not necessarily be a feasible solution to the MC problem on the original graph instance G . Moreover, in the worst case, $OPT(G'')$ may include only those vertices that correspond to larger demand values in the original graph. This observation and an argument similar to Lemma 1 can be used to prove Lemma 2. For conciseness, we omit the details. We have the following result for the approximation ratio.

Theorem 4. *The Smallest Demand First or SDF algorithm is a 4β -approximation algorithm for the MC problem, where $\beta > 1$ is the approximation ratio of the minimum WVC algorithm. Moreover, the algorithm runs in $O(mn)$ time, where n is the number of vertices and m is the number of edges in the graph.*

The proof for Theorem 4 can be found in the Appendix. It is clear from Theorem 4 that the SDF algorithm guarantees the same approximation ratio as

the deterministic algorithm of Xu et al. [47] but runs slower. We now evaluate the practical efficiency of the proposed approaches by executing them on real vehicular road-network data.

5 Empirical Evaluation

In this section, we evaluate the performance of the proposed algorithms by implementing them in Matlab and executing them on a multi-core desktop computer. For our experiments, we construct the input graph instance using real road-traffic data (intersections, roads and bi-directional AADT traffic intensities) from the official transportation databases for Florida [17] and Virginia [46]. The results are outlined in Table 1.

Table 1. Performance of the proposed Mix Cover algorithms on real road traffic data.

		SMALL SIZE GRAPH 25% of municipalities			MEDIUM SIZE GRAPH 65-85% of municipalities			FULL SIZE GRAPH Entire State		
Tot. # of v / e	Florida	2557 / 2640			6326 / 6960			7557 / 8310		
	Virginia	2408 / 2514			5726 / 6728			5881 / 6952		
# of v in the MC solution	A 1 - Florida	1243	1267	1238	2891	2990	2909	3481	3600	3545
	A 2 - Florida	1217	1244	1216	2863	2949	2877	3452	3502	3452
	A 1 - Virginia	1346	1373	1350	3328	3404	3345	3523	3592	3532
	A 2 - Virginia	1376	1386	1364	3376	3426	3374	3550	3607	3551
Ratio MC solution obj. fct. / naïve obj. fct.	A 1 - Florida	0.51	0.40	0.48	0.49	0.38	0.46	0.49	0.38	0.46
	A 2 - Florida	0.45	0.35	0.42	0.43	0.34	0.40	0.43	0.34	0.40
	A 1 - Virginia	0.82	0.79	0.81	0.82	0.79	0.81	0.82	0.79	0.81
	A 2 - Virginia	0.79	0.76	0.78	0.79	0.77	0.79	0.80	0.77	0.79
Duration of the MC simulation [sec]	A 1 - Florida	13.08	15.84	16	54.56	58.52	65.48	68.43	71.84	83.73
	A 2 - Florida	14.89	18.26	18.37	59.59	62.16	71.1	77.34	77.94	93.36
	A 1 - Virginia	12.78	15.08	15.58	43.02	51.22	54.98	42.47	49.13	50.7
	A 2 - Virginia	13.45	16.05	16.53	46.68	54.47	57.58	46.17	51	54

For each state, we considered three different sizes of the respective road network graphs: a small graph that corresponds to 25% of the total number of municipalities, a medium (65-85%) and a full state graph. For each such graph, we evaluated the performance of the proposed algorithms for three vertex weight distributions, namely, constant, uniform and positive Gaussian. The constant distribution assigns the same weight ($=1$) to all vertices, the uniform draws the weights uniformly at random from the interval $[1,100]$, whereas the Gaussian has an expected value of 50 and a standard deviation of 10. Based on these parameters and the traffic intensities (or vertex demands) for each state, we measured the ratio between the MC solution objective function and the worst-case (naïve) solution (which includes all vertices of the graph in the vertex cover), the number of vertices in the MC solution, the individual vertex capacities and the duration of the simulation. The values in Table 1 are averaged over 100 runs.

The results confirm that, as predicted by the analytical evaluation, SDF (A2) performs consistently better than LDF (A1) for all graph sizes. Compared to the

naïve solution, the proposed algorithms achieve a lower mix-zone cost, as low as 34% of the trivial solution cost. For all combinations of parameters, the uniform distribution achieves the best (lowest) objective function ratio, followed by the Gaussian and the constant distributions. The uniform distribution, which assigns (on average) the same weight to an equal number of vertices, makes it easier to determine feasible capacities to vertices that have a lower weight, while still covering all edges of the graph. In the Gaussian scenario, most of the weights will be close to the mean, and thus the search for the vertices that minimize the weighted cost will be more complex, leading to a worse solution and more demanding in terms of computation time. In the case where all vertices have the same weights, there are no chances of finding a feasible solution consisting of vertices with a lower weight than others. Hence, the ratio of the MC solution to that of the naïve solution is the worst in this scenario.

Depending on size of the graph and the respective demands, the number of mix-zones to be deployed is between 46% (Florida) and 58% (Virginia) of the total number of vertices. In Florida, SDF performs slightly better than LDF as it requires a smaller number of mix-zones. Although the differences amount to 2-3% (up to 102 fewer mix-zones), such result is consistent across all graph sizes. In Virginia, on the contrary, LDF performs slightly better than SDF (up to 30 fewer mix-zones). This indicates that, although relatively small, the performance of the two algorithms are influenced by the road network topology, and further investigations are required in order to determine the effects of the road topology on the performance of the proposed algorithms.

Intuitively, as the traffic patterns evolve during the day in each region, such algorithms would be executed multiple times per day in order to adapt the solutions to the traffic intensities throughout the day. Regarding the execution efficiency, the experimental results show that a feasible solution to the MC problem can be determined in 13 sec (small graph) and 94 sec (full State graph), which is a reasonable requirement in case such computations are done in a dynamic fashion multiple times per day.

In order to avoid unbounded solutions in the LP formulation, we had to reduce the graph size (and thus the number of constraints) of the road network. Considering a reduced (Florida) graph with 515 vertices, we obtained a ratio of 0.24 between the objective function of the MC solution with respect to the naïve one, which is a better result than LDF and SDF, but the fraction of intersections with mix-zones to the total number of intersections increased to 97%. For such a small graph, the LP required between 29 seconds (constant distribution) and 66 seconds (Gaussian distribution), which corresponds to two and four times the requirement of LDF and SDF, respectively, with the same weight distributions. Similar relative differences were obtained when increasing the number of vertices from 515 to 1024, except that the durations grew by a factor of 20 as compared to LDF and SDF. The results suggest that the LP formulation yields on average better (lower) costs for the mix-zone deployment, at the expense of a significant increase in computation time and number of mix-zones. Hence, the LP approach

appears to be better suited for smaller graphs with a lower intersection/road density, such as peripheral and rural areas.

6 Conclusion

We addressed the problem of optimizing mix-zone placement in pervasive networking applications by formulating it as a graph-based optimization problem, referred to as the Mix Cover or MC problem. We proposed three algorithms to solve the MC problem: the first algorithm is based on a LP relaxation of the problem and the remaining two approaches take advantage of a “divide and conquer” strategy proposed by Xu et al. [47]. We proved important analytical results, such as the solution quality and running-time guarantees, for the proposed approaches. In order to shed light on their feasibility in a realistic pervasive network setting, we performed extensive experimental evaluation of the proposed approaches with real road network and traffic data. Experimental results confirmed the analytical results and also showed that these approaches can compute an approximate mix cover, even for fairly large road networks, in a reasonable amount of time using standard computing resources.

References

1. <http://en.wikipedia.org/wiki/Lovegetty>.
2. <http://en.wikipedia.org/wiki/Bluedating>.
3. <http://pleaserobme.com/>.
4. A. Ahtiainen, K. Kalliojarvi, M. Kasslin, K. Leppanen, A. Richter, P. Ruuska, and C. Wijting. Awareness networking in wireless environments: Means of exchanging information. *IEEE Vehicular Technology Magazine*, September 2009.
5. T. Alpcan and S. Buchegger. Security games for vehicular networks. *IEEE Transactions on Mobile Computing*, 2011.
6. B. Aspval and R. Stone. Khachiyan’s linear programming algorithm. *Journal of Algorithms*, 1(1):1 – 13, 1980.
7. R. Bar-Yehuda and S. Even. A linear time approximation algorithm for the weighted vertex cover algorithm. *Journal of Algorithms*, 1981.
8. A. R. Beresford and F. Stajano. Location privacy in pervasive computing. *Pervasive Computing, IEEE*, 2(1):46–55, 2003.
9. A. R. Beresford and F. Stajano. Mix zones: User privacy in location-aware services. In *PerCom Workshop*, 2004.
10. S. Buchegger, D. Schiöberg, L-H Vu, and A. Datta. Peerson: P2P social networking: early experiences and insights. In *EuroSys SNS Workshop*, pages 46–52, 2009.
11. L. Buttyán, T. Holczer, and I. Vajda. On the effectiveness of changing pseudonyms to provide location privacy in vanets. In *ESAS*, 2007.
12. L. Buttyán, T. Holczer, A. Weimerskirch, and W Whyte. Slow: A practical pseudonym changing scheme for location privacy in vanets. In *IEEE VNC*, 2009.
13. D. Chaum. Untraceable electronic mail, return addresses and digital pseudonyms. *Comm. ACM*, 24(2):84–88, 1981.
14. D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *J. Cryptology*, 1(1):66–75, 1988.

15. G. B. Dantzig. *Linear Programming and Extensions*. Princeton Press, 1963.
16. I. Dinur and S. Safra. On the hardness of approximating minimum vertex-cover. *Annals of Mathematics*, 2005.
17. Florida State traffic data. <http://www.dot.state.fl.us/planning/statistics/gis/trafficdata.shtm>.
18. J. Freudiger, M. Raya, M. Felegyhazi, P. Papadimitratos, and J-P. Hubaux. Mix zones for location privacy in vehicular networks. In *Win-ITS*, 2007.
19. J. Freudiger, R. Shokri, and J-P. Hubaux. On the optimal placement of mix zones. In *PETS*, 2009.
20. J. Freudiger, R. Shokri, and J-P. Hubaux. Evaluating the privacy risk of location-based services. In *Financial Cryptography*, 2011.
21. S. Gaonkar, J. Li, R. R. Choudhury, L. P. Cox, and A. Schmidt. Micro-blog: sharing and querying content through mobile phones and social participation. In *MobiSys*, pages 174–186, 2008.
22. M. R. Garay and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
23. E. Giordano, A. Tomatis, A. Ghosh, G. Pau, and M. Gerla. C-VeT: An Open Research Platform for Vanets: Evaluation of Peer to Peer Applications in Vehicular Networks. In *IEEE VTC*, 2008.
24. O. Goldreich. *Computational Complexity: A Conceptual Perspective*. 2008.
25. P. Golle and K. Partridge. On the anonymity of home/work location pairs. In *Pervasive '09*, 2009.
26. T. F. Gonzalez. A simple LP-free approximation algorithm for the minimum weight vertex cover problem. *Information Processing Letters*, 54(3):129–131, 1995.
27. R. Hassin and A. Levin. The minimum generalized vertex cover problem. *ACM Trans. Algorithms*, 2, January 2006.
28. D. Hochbaum and A. Levin. The Multi-integer Set Cover and the Facility Terminal Cover Problem. *Networks*, 53:63–66, January 2009.
29. L. Huang, K. Matsuura, H. Yamane, and K. Sezaki. Enhancing wireless location privacy using silent period. In *IEEE WCNC*, 2005.
30. L. Huang, H. Yamane, K. Matsuura, and K. Sezaki. Towards modeling wireless location privacy. In *PETS*, 2006.
31. M. Humbert, M. H. Manshaei, J. Freudiger, and J-P. Hubaux. Tracking games in mobile networks. In *GameSec*, 2010.
32. R.M. Karp. *Complexity of Computer Computations*. Plenum Press, 1972.
33. M. Khiabani. Metro-sexual. <http://bit.ly/theranMetroSexual>, 2009.
34. V. Klee and G. J. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities*, volume III, pages 159–175. Academic Press, 1972.
35. M. Li, K. Sampigethaya, L. Huang, and R. Poovendran. Swing & swap: user-centric approaches towards maximizing location privacy. In *WPES*, 2006.
36. N. Megiddo. On the complexity of linear programming. In *Advances in Economic Theory, Fifth World Congress*, pages 225–268. Cambridge University Press, 1987.
37. C.J. Merlin and W.B. Heinzelman. A study of safety applications in vehicular networks. In *MASS '05*, 2005.
38. B. Palanisamy and L. Liu. Mobimix: Protecting location privacy with mix zones over road networks. In *ICDE '11*, 2011.
39. E. Paulos and E. Goodman. The familiar stranger: anxiety, comfort, and play in public places. In *CHI*, pages 223–230, 2004.
40. A. Pfizmann and M. Köhntopp. Anonymity, unobservability, and pseudonymity - a proposal for terminology. In *International workshop on Designing privacy enhancing technologies: design issues in anonymity and unobservability*, 2001.

41. Rhiain. Nokia instant community gets you social. <http://conversations.nokia.com/2010/05/25/nokia-instant-community-gets-you-social/>.
42. K. Sampigethaya, L. Huang, M. Li, R. Poovendran, K. Matsuura, and K. Sezaki. CARAVAN: Providing location privacy for VANET. In *ESCAR*, 2005.
43. E. Schoch, F. Kargl, T. Leinmiller, S. Schlott, and P. Papadimitratos. Impact of pseudonym changes on geographic routing in vanets. In *ESAS*, 2006.
44. R. Shokri, J. Freudiger, M. Jadliwala, and J-P. Hubaux. A distortion-based metric for location privacy. In *ACM WPES*, 2009.
45. R. Shokri, G. Theodorakopoulos, J-Y. Le Boudec, and J-P. Hubaux. Quantifying location privacy. In *IEEE S&P*, 2011.
46. Virginia State traffic data. http://www.virginiadot.org/info/2009_traffic_data.asp.
47. G. Xu, Y. Yang, and J. Xu. Linear Time Algorithms for Approximating the Facility Terminal Cover Problem. *Networks*, 50:118–126, August 2007.
48. Q. Xu, T. Mak, J. Ko, and R. Sengupta. Vehicle-to-vehicle safety messaging in dsr. In *VANET*, 2004.

Appendices

Proof (Proof for Lemma 1). The first part of the lemma is straightforward. As the vertex and edge sets of both G and G' are the same, a cover for G' (that covers all edges of G') is also a cover for G . Moreover, as the edge demands are rounded off to the largest and then to the closest power of 2, the selected capacity of the vertices for a solution (or mix cover) in G' will always be greater than the demands of the corresponding edges in G . Thus, a mix cover for G' is also a feasible mix cover for G .

Now, let us prove the second part. Let $V_{OPT(G)}$ be the set of vertices of the optimal solution $OPT(G)$ for the MC problem on the graph instance G and let $C_{OPT(G)} = \{c(v_i) | v_i \in V_{OPT(G)}\}$ be the capacities assigned to each vertex in the optimal solution. Consider a solution S such that it has the same set of vertices as $V_{OPT(G)}$ and with capacities $C_S = \{2(c(v_i) + \alpha) | v_i \in S\}$, where α is as defined above. We can see that (S, C_S) is always a feasible solution to G' . This is because, firstly, we always select the larger demand value for each edge e . Thus, even in the worst case, where all vertices $v_i \in V_{OPT(G)}$ in the optimal solution $OPT(G)$ are such that $d_e^{v_i} < d_e^{v_j}, \forall e \equiv (v_i, v_j) \in E$, capacities selected in C_S will always overcome the difference in demands. Secondly, the rounded demands of each edge e is only at most twice that of the original (larger) demand. Thus,

$$\begin{aligned}
 OPT(G') &\leq \sum_{v_i \in S} 2(c(v_i) + \alpha) \\
 &\leq \sum_{v_i \in S} 2\alpha c(v_i) \leq 2\alpha OPT(G) \quad \square
 \end{aligned}$$

Proof (Proof for Theorem 2). For the sake of convenience, let us denote the IP formulation of the MC problem as IP-MC and its LP relaxation as LP-MC. It is easy to see that any optimal solution OPT to IP-MC is also a feasible solution to the LP-MC and has the same objective function value. Moreover, LP-MC is

indeed a relaxation of IP-MC and an optimal solution $(\bar{x}, \{\bar{z}_e\})$ to LP-MC is a feasible fractional solution to IP-MC. Thus, we can see that the value of the objective function (as the objective functions for both the formulations are the same) of an optimal solution $(\bar{x}, \{\bar{z}_e\})$ to LP-MC is at most that of the optimal solution OPT to IP-MC. Now, given the optimal solution $(\bar{x}, \{\bar{z}_e\})$ to LP-MC, we know that for any $e \equiv (u, v) \in E$, as $z_e^u + z_e^v \geq 1$, at least one of the following $z_e^u \geq \frac{1}{2}$ or $z_e^v \geq \frac{1}{2}$ is true. Let us apply the following transformation δ to $(\bar{x}, \{\bar{z}_e\})$: If $z_{e,i} \geq \frac{1}{2}$, for any e and i , then $\delta(z_{e,i}) = 1$ and if $z_{e,i} < \frac{1}{2}$ then put $\delta(z_{e,i}) = 0$. Also, $\delta(x_i) = 2x_i$ if for any $i \in V$ there is $\delta(z_{e,i}) = 1$.

It is easy to see that $\delta(\bar{x}, \{\bar{z}_e\})$ is a feasible solution to the IP-MC problem. Moreover, the linearity of the objective function guarantees that the objective function value (or the total weighted cost) of $\delta(\bar{x}, \{\bar{z}_e\})$ is at most twice the objective function value of $(\bar{x}, \{\bar{z}_e\})$. As the cost of $(\bar{x}, \{\bar{z}_e\})$ is at most OPT , the cost of $\delta(\bar{x}, \{\bar{z}_e\})$ is at most two times the cost of OPT , i.e., $\delta(\bar{x}, \{\bar{z}_e\}) \leq 2 \cdot OPT$. Thus, $\delta(\bar{x}, \{\bar{z}_e\})$ is a 2-approximation of the MC problem. Moreover, as LP-MC can be solved in polynomial time [6], the proof follows. \square

Proof (Proof for Theorem 3). Let $WVC-2Approx(G_k)$ denote the output (overall minimum weight) of applying a β -approximation minimum WVC algorithm to the subgraph G_k of G' . If $OPT(G_k)$ is the corresponding optimal solution, then we have the following inequality:

$$\begin{aligned} WVC-2Approx(G_k) &\leq \beta OPT(G_k) \\ OPT(G_k) &\geq \frac{1}{\beta} WVC-2Approx(G_k) \end{aligned} \quad (1)$$

From Lemma 2 of [47] we know that,

$$OPT(G') \geq \frac{1}{2} \sum_{k=0}^K 2^k OPT(G_k) \quad (2)$$

where K is max. value of the exponent after the rounding. From Lemma 1,

$$OPT(G') \leq 2\alpha OPT(G) \quad (3)$$

From (1) and (2), we have

$$\begin{aligned} OPT(G') &\geq \frac{1}{2} \sum_{k=0}^K \frac{2^k}{\beta} WVC-2Approx(G_k) \\ &\geq \frac{1}{2\beta} \sum_{k=0}^K 2^k WVC-2Approx(G_k) \end{aligned}$$

Combining the above inequality with Eqn. 3 we have,

$$\begin{aligned} 2\alpha OPT(G) &\geq \frac{1}{2\beta} \sum_{k=0}^K 2^k WVC-2Approx(G_k) \\ \sum_{k=0}^K 2^k WVC-2Approx(G_k) &\leq 4\alpha\beta OPT(G) \end{aligned} \quad (4)$$

The left-hand side of the inequality in (4) clearly denotes the objective function computed from the output of the LDF algorithm. From this inequality, it is clear that the LDF algorithm is a $4\alpha\beta$ -approximation of the MC problem.

Now, let us observe the time complexity of the LDF algorithm. Tasks such as determining the larger demand per edge, rounding the demand value and assigning capacities can be completed in linear time, in the worst-case. Moreover, Bar-Yehuda and Even [7] showed that a 2-approximation can be obtained for a WVC problem in linear time. Thus, the LDF algorithm runs in linear time. \square

Proof (Proof for Theorem 4). We use a similar argument that we use to prove Theorem 3. From Eqn. 1 in the proof of Theorem 3 we know that:

$$OPT(G_k) \geq \frac{1}{\beta} \text{WVC-2Approx}(G_k)$$

where, β is the approximation ration of the weighted vertex cover algorithm WVC-2Approx and G_k is the induced subgraph of G'' with edge demands 2^k . Now, let K'' be the maximum value of the exponent (of 2) after the rounding the shortest demands on each edge. From Lemma 2 of [47] we know that, $OPT(G'') \geq \frac{1}{2} \sum_{k=0}^{K''} 2^k OPT(G_k)$. From Lemma 2 we have, $OPT(G'') \leq \frac{2}{\alpha} OPT(G)$

From the above we have,

$$\begin{aligned} \frac{2}{\alpha} OPT(G) &\geq \frac{1}{2} \sum_{k=0}^{K''} 2^k OPT(G_k) \\ 4OPT(G) &\geq \alpha \sum_{k=0}^{K''} 2^k OPT(G_k) \\ 4\beta OPT(G) &\geq \alpha \sum_{k=0}^{K''} 2^k \text{WVC-2Approx}(G_k) \\ 4\beta OPT(G) &\geq \sum_{k=0}^{K''} (2^k + \alpha) \text{WVC-2Approx}(G_k) \end{aligned} \tag{5}$$

The right-hand side of (5) clearly denotes an upper bound on the objective function computed by the SDF algorithm. From this inequality, it is clear that the SDF algorithm is a 4β -approximation algorithm for the MC problem.

Now, let us observe the time complexity of the SDF algorithm. The SDF algorithm has an additional step, as compared to the LDF algorithm, which is the largest demand determination on each vertex. It is easy to see that this step, in the worst case, takes $O(n^2)$ additional steps and thus one order of time more than the LDF algorithm. \square